UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/589,239	01/03/2007	Ivan Boule	1483-32	6941

23117 7590 05/27/2009
NIXON & VANDERHYE, PC
901 NORTH GLEBE ROAD, 11TH FLOOR
ARLINGTON, VA 22203

EXAMINER	
SADLER, NATHAN	
ART UNIT	PAPER NUMBER
2189	
MAIL DATE	DELIVERY MODE
05/27/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/589,239	BOULE ET AL.	
	Examiner	Art Unit	
	Nathan Sadler	4123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *2/14/05*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-47* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-47* is/are rejected.

7)☒ Claim(s) *39 and 47* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *8/14/06* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☒ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☒ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *8/14/06*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Information Disclosure Statement

1. The information disclosure statement filed on August 14, 2006 fails to comply with 37 CFR 1.98(a)(2), which requires a legible copy of each cited foreign patent document; each non-patent literature publication or that portion which caused it to be listed; and all other information or that portion which caused it to be listed. It has been placed in the application file, but the information referred to therein has not been considered.

Specification

2. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

3. Claim 39 is objected to under 37 CFR 1.75(c), as being of improper dependent form for failing to further limit the subject matter of a previous claim. Applicant is required to cancel the claim(s), or amend the claim(s) to place the claim(s) in proper dependent form, or rewrite the claim(s) in independent form. Claim 39 merely recites the preamble of claim 17 and is dependent on claim 17.

4. Claim 47 is objected to under 37 CFR 1.75(c) as being in improper form because a multiple dependent claim should refer to other claims in the alternative only. See MPEP § 608.01(n). Accordingly, the claim 47 has not been further treated on the merits.

5. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction

of the following is required: in claim 1, "the allocation" in line 1, "the lowest" in line 5, "the same size" in line 5, and "the same" in line 8; in claim 2, "the next higher level" in line 2; in claim 3, "the sizes" in line 2; in claim 4, "the size difference" in line 2; in claim 5, "the lowest" in line 2; in claim 9, "the state" in line 2; in claim 10, "the size" in line 2, "the most significant bit" in lines 4-5, and "the lowest" in line 6; in claim 11, "the operation result" in line 4; in claim 12, "the state" in line 4; in claim 13, "the next" in line 2; in claim 17, "the size" in line 5, "the allocation" in line 6, and "the same size" in line 7; in claim 18, "the size difference" in lines 1-2; in claim 19, "the state" in line 2; in claim 28, "the order" in line 2; in claim 29, "the state" in line 3; in claim 32, "the remaining portion" in line 6; in claim 33, "the state" in line 4 and "the memory address" in line 5; in claim 34, "the header" in lines 1-2, "the memory address" in line 3, and "the availability" in line 4; in claim 35, "the availability" in line 5 and "the state" in line 6; in claim 36, "the memory address" in line 2; and in claim 38, "the state" in line 4.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claims 1-16, 35-37, and 40-47 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

8. Claims 1 and 35 recites "their size" in line 3 and lines 2-3 respectively. It is indefinite what "their" is referring to in these cases. It could be referring to the segments, the data memory, or the different levels.

9. Claim 1 recites "depending on" in line 9. The phrase "depending on" renders the claim indefinite because it is unclear whether the limitation "allocating a free segment" is part of the claimed invention.

Claim Rejections - 35 USC § 101

10. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

11. Claims 1-47 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. None of these claims are tied to a particular machine, nor do they bring about a particular transformation of a particular article. Claims 1, 17, 27, 35, and 38 all recite "a data memory". Claims 40, 44, and 45 all recite "a computer". Claim 46 recites "a computer system". Claim 47 recites "a processor". All of these limitations are insufficient to tie the methods recited in the claims to a particular machine. Both "a data memory" and "a processor" are elements inherently found in a generic computer system. Similarly, "a computer" and "a computer system" are simply different ways to refer to a generic computer system. The mere recitation of a generic computer system is not sufficient to a method to a particular machine.

Claim Rejections - 35 USC § 102

12. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

13. Claims 1, 27-34, and 40-43 are rejected under 35 U.S.C. 102(b) as being anticipated by Wilson et al. ("Dynamic Storage Allocation: A Survey and Critical Review").

14. In regards to claim 1, Wilson teaches a method of processing requests for the allocation of a memory block of a data memory, wherein segments of the data memory are allocated to different levels according to their size, the method comprising the steps of:

15. (a) receiving a request for the allocation of a memory block ("the a program can request a block of memory to store a program object", page 1, paragraph 4);

16. (b) determining the lowest of said levels containing a segment of the same size as or larger than the requested memory block ("When a request is serviced, the free list for the appropriate size is used to satisfy the request.", page 51, paragraph 6);

17. (c) determining, in the level determined in step (b), the availability of a free segment of a size the same as or larger than the requested memory block ("the free list for the corresponding size class is searched for a block at least large enough to hold it.", page 53, paragraph 3); and

18. (d) depending on the determination in step (c), allocating a free segment ("An allocated block", page 1, paragraph 4).

19. In regards to claim 27, Wilson teaches a method of managing a data memory comprising memory segments of different sizes for allocation in response to a memory allocation request, the method comprising:

20. creating a first doubly linked list of consecutive memory segments irrespective of size and status (free, allocated) ("Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas. Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7); and

21. creating a second doubly linked list of free memory segments ("Many systems use doubly-linked linear lists", page 40, paragraph 4) of the same size ("One of the simplest allocators uses an array of free lists, where each list holds free blocks of a particular size", page 51, paragraph 6).

22. In regards to claim 28, Wilson further teaches that memory segments in the first doubly linked list are arranged in the order of associated memory addresses ("Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas. Each block of memory has a both header and a 'footer' field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7).

39. In regards to claim 29, Wilson further teaches determining the state of memory segments adjacent to the memory segment to be freed using the first doubly linked list ("Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas., page 39, paragraph 7);

40. merging the memory segment to be freed with free adjacent memory segments ("Many allocators that support general coalescing are implemented using boundary tags

(due to Knuth [Knu73]) to support the coalescing of free areas.", page 39, paragraph 7); and

41. updating the first and second doubly linked lists accordingly ("Adjacent free areas are merged to form larger free blocks", page 39, paragraph 7).

23. In regards to claim 30, Wilson further teaches that each second doubly linked list is a LIFO (Last In First Out) list ("it appears that address-ordered first fit may cause significantly less fragmentation than LIFO-ordered first fit", page 44, paragraph 6).

24. In regards to claim 31, Wilson further teaches updating the second doubly linked list upon allocation of a memory segment upon request ("When a request is serviced, the free list for the appropriate size is used to satisfy the request.", page 51, paragraph 6).

25. In regards to claim 32, Wilson further teaches allocating a portion of the determined segment large enough to satisfy the request ("the free list for the corresponding size class is searched for a block at least large enough to hold it.", page 53, paragraph 3);

26. providing the remaining portion as a new free memory segment ("An allocated block", page 1, paragraph 4); and

27. updating the first and second doubly linked lists accordingly ("Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7, "One of the simplest allocators uses an array of free lists, where each list holds free blocks of a particular size", page 51, paragraph 6).

28. In regards to claim 33, Wilson further teaches that each segment is associated with a header, thereby to form the first doubly linked list, each header including information indicative of the size of the associated segment, information indicative of the state (free, allocated) of the associated segment, and a pointer indicative of the memory address of the previous segment ("Many allocators that support general coalescing are implemented using boundary tags (due to Knuth [Knu73]) to support the coalescing of free areas. Each block of memory has a both header and a "footer" field, both of which record the size of the block and whether it is in use.", page 39, paragraph 7).

29. In regards to claim 34, Wilson further teaches that the header associated with each free segment of a given size further includes a pointer indicative of the memory address of a previous and/or subsequent free segment of the same size, depending on the availability of a previous and/or subsequent free segment of the same size and in accordance with the order of free segments of the same size in the LIFO list ("For allocators using free lists or indexing trees to keep track of free blocks, the list or tree nodes are generally embedded in the free blocks themselves.", page 40, paragraph 3).

30. In regards to claim 40, Wilson further teaches an operating system for a computer ("operating systems", page 7, paragraph 2).

31. In regards to claim 41, Wilson further teaches that the operating system is a realtime operating system ("real-time and/or concurrent systems", page 2, footnote 5).

32. In regards to claim 42, Wilson further teaches a system adapted to perform the method described above at task level (inherent to not using an interrupt, "Another

possibility would be to use a timer interrupt, but this is quite awkward.", page 65, footnote 83).

33. In regards to claim 43, Wilson further teaches a system adapted to perform the method described above at interrupt level ("Another possibility would be to use a timer interrupt, but this is quite awkward.", page 65, footnote 83).

34. Claims 1-7, 9-11, 17-18, 23-26, 35-39, 44-47 are rejected under 35 U.S.C. 102(b) as being anticipated by McMahon et al. (US 5,784,699).

35. In regards to claim 1, McMahon teaches a method of processing requests for the allocation of a memory block of a data memory, wherein segments of the data memory are allocated to different levels according to their size, the method comprising the steps of:

36. (a) receiving a request for the allocation of a memory block ("software programs generate requests", abstract);

37. (b) determining the lowest of said levels containing a segment of the same size as or larger than the requested memory block (step 110, figure 2);

38. (c) determining, in the level determined in step (b), the availability of a free segment of a size the same as or larger than the requested memory block (step 120, figure 2); and

39. (d) depending on the determination in step (c) (step 130, figure 2), allocating a free segment (step 135, figure 2).

40. In regards to claim 2, McMahon further teaches e) repeating steps (c) and (d) for the next higher level if no free segment of a size the same as or larger than the requested memory block has been found in step (c) (step 140, figure 2); and

41. (f) repeating step (e) until a free segment has been allocated or there is no next level (the no arrow of step 150, figure 2).

42. In regards to claim 3, McMahon further teaches that each level is associated with a different granule size to the power of two ("For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.", Col. 2, lines 42-45), and the sizes of memory segments allocated to a level are related to the granule size of the respective level ("For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.", Col. 2, lines 42-45).

43. In regards to claim 4, McMahon further teaches that the granule size associated with a level defines the size difference between memory segments allocated to that level ("For example, the size based free lists technique may utilize a power of two selection technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.", Col. 2, lines 42-45).

44. In regards to claim 5, McMahon further teaches that step (a) further comprises rounding the requested memory block to the lowest granule size before performing steps (b) to (d) ("The block size rounding technique attempts to reduce fragmentation of memory by rounding all requests up to some minimum size.", Col. 2, lines 18-20).

45. In regards to claim 6, McMahon further teaches that each level is associated with a table of pointers indicative of memory addresses of free memory segments of a size allocated to the respective level ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135", Col. 5, lines 32-35).

46. In regards to claim 7, McMahon further teaches that step (d) comprises returning a pointer to the allocated free segment ("In response, the dynamic memory allocator 50 returns a pointer to a memory block that fulfills the memory request", Col. 5, lines 1-3).

47. In regards to claim 9, McMahon further teaches that a bitmap is indicative of the state of memory segments (free, allocated), the bitmap comprising a root bitmap, each bit of the root bitmap being indicative of whether or not an associated one of said levels contains at least one free segment (master bit map index, 340, figure 3C), and wherein step (b) further comprises determining from the root bitmap said lowest level containing a segment of a size the same as or larger than the requested memory block ("From the master bit map index 340, the dynamic memory allocator determines that the next appropriate non-empty group of free lists is group 6.", Col. 8, lines 2-4).

48. In regards to claim 10, McMahon further teaches that step (a) comprises receiving a binary data set indicative of the size of the requested memory block ("For example, in the C programming language, the C run time library of many systems includes the functions malloc(size)", Col. 1, lines 37-39), wherein each bit of the binary data set is associated with an entry of a lookup table associated with one of said levels ("For example, the size based free lists technique may utilize a power of two selection

technique to designate 16 byte, 32 byte, 64 byte, 128 byte, 256 byte, etc., blocks as the predetermined block sizes.", Col. 2, lines 42-46), and step (b) comprises determining the most significant set bit of the binary data set, and determining from the entry of the lookup table associated with the most significant set bit the lowest of said levels containing a segment of a size the same as or larger than the requested memory block (step 110, figure 2).

49. In regards to claim 11, McMahon further teaches that each mask of a set of predetermined masks is associated with one of said levels, and step (c) further comprises performing a logic operation on the mask associated with the lowest level determined in step (b) and said binary data set, wherein the operation result is an index to bits of the bitmap indicative of the state of a segment of a size the same as or larger than the requested memory block ("Then, the dynamic memory allocator, utilizing the group 1 bit map index 360, masks off the lower bit flags, corresponding to free lists 1-6, to search for a memory block greater than or equal to a 112 byte block. Because bit flags 7-32 are all set to "0", which indicates that there are no available memory blocks in free lists 7-32, the dynamic memory allocator returns to the master bit map index 340.", Col. 7, line 62 - Col. 8, line 2).

50. In regards to claim 17, McMahon teaches a method of managing a data memory, the method comprising:

51. defining a number of levels of the data memory (free lists, Col. 4, line 33);

52. defining a different granule size for each level (bin size, Col. 4, line 49);

53. defining a different range of a plurality of different sizes of memory segments for each level, wherein the size of each memory segment is related to the granule size of the respective level ("The rounding factor is defined as the maximum amount of memory a request is rounded to execute a dynamic memory allocation operation.", Col. 4, lines 50-53), and wherein a request for the allocation of a memory block is processable by determining a level containing segments of the same size as or larger than the requested memory block (step 120, figure 2), and allocating a free segment of a size the same as or larger than the requested memory block in that level (step 135, figure 2).

54. In regards to claim 18, McMahon further teaches that the granule size defines the size difference between memory segments in each level ("The rounding factor is defined as the maximum amount of memory a request is rounded to execute a dynamic memory allocation operation.", Col. 4, lines 50-53).

55. In regards to claim 23, McMahon further teaches generating a table of pointers for each level indicative of memory addresses of free memory segments of a size associated with the respective level ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135.", Col. 5, lines 32-35).

56. In regards to claim 24, McMahon further teaches each bit of the third stage bitmaps is associated with an entry in the tables of pointers ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135.", Col.

5, lines 32-35, "The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size.", Col. 7, lines 26-28).

57. In regards to claim 25, McMahon further teaches generating a lookup table, wherein each entry of the lookup table is associated with a bit of a binary data set indicative of the size of the requested memory block and indicative of one of said levels (See Table 2, Col. 14).

58. In regards to claim 26, McMahon further teaches generating a set of masks, wherein each of the set of masks is associated with one of said levels, and wherein a logical operation of a binary data set indicative of the size of the requested memory block and the mask associated with a level containing segments of the same size as or larger than the requested memory block results in an index to a segment of a size the same as or larger than the requested memory block in that level ("Then, the dynamic memory allocator, utilizing the group 1 bit map index 360, masks off the lower bit flags, corresponding to free lists 1-6, to search for a memory block greater than or equal to a 112 byte block.", Col. 7, lines 62-65).

59. In regards to claim 35, McMahon teaches a method of managing a data memory, the method comprising:

60. allocating free segments of the data memory to different levels according to their size (step 120, figure 2); and

61. providing a bitmap comprising different stages (see figure 3C), wherein the bits of one stage are indicative of the availability of free segments in said levels (master bit map index and arrows, figure 3C), and the bits of another stage are indicative of the

state and/or size and/or location of individual segments (group bit map indices, 350 and 360, figure 3C).

62. In regards to claim 36, McMahon further teaches the bits of one stage are associated with pointers indicative of the memory address of free segments ("If a memory block is available for the appropriate bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135.", Col. 5, lines 32-35, "The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size.", Col. 7, lines 26-28).

63. In regards to claim 37, McMahon further teaches updating the bitmap to reflect the allocation or release of memory segments ("a bit map index is maintained to identify available memory blocks", Col. 4, lines 38-39).

64. In regards to claim 38, McMahon teaches a method of managing a data memory, including freeing and allocating segments of the data memory, the method comprising, when freeing a memory segment:

65. determining the state of memory segments adjacent to the memory segment to be freed ("To free a large block of memory, adjacent blocks, marked as free memory blocks, are merged together, if possible, through use of the double linked list.", Col. 11, lines 34-36); and

66. merging the memory segment to be freed with free adjacent memory segments ("To free a large block of memory, adjacent blocks, marked as free memory blocks, are

merged together, if possible, through use of the double linked list.", Col. 11, lines 34-36).

67. In regards to claim 39, McMahon further teaches a method of managing a data memory (title).

68. In regards to claim 44, McMahon further teaches a computer program adapted to perform the method of claim 1 when operated on a computer ("The dynamic memory allocation system may be implemented in either hardware or software.", Col. 16, lines 13-14).

69. In regards to claim 45, McMahon further teaches a storage medium having stored thereon a set of instructions, which when executed by a computer, performs the method of claim 1 ("Prior to loading into a general purpose computer system, the dynamic memory allocation system software may reside as encoded information on a computer readable medium", Col. 16, lines 17-20).

70. In regards to claim 46, McMahon further teaches a computer system programmed to perform the method of claim 1 ("Prior to loading into a general purpose computer system, the dynamic memory allocation system software may reside as encoded information on a computer readable medium", Col. 16, lines 17-20).

71. In regards to claim 47, McMahon further teaches a processor (processor unit, 1005, figure 6) arranged to perform the method of claim 1.

Claim Rejections - 35 USC § 103

72. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

73. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Wilson et al. ("Dynamic Storage Allocation: A Survey and Critical Review") in view of RedHat ("MALLOC(3)").

74. Wilson teaches the method of claim 1 as discussed above. Wilson fails to teach returning a null pointer if no free segment is allocated. RedHat teaches returning a null pointer if no free segment is allocated ("NULL if the request fails", RETURN VALUES). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine Wilson and RedHat to return a null pointer if no free segment is allocated because RedHat describes the function of malloc() which Wilson suggests as an example of an allocator (paragraph 2, page 39).

75. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of RedHat ("MALLOC(3)").

76. McMahon teaches the method of claim 1 as discussed above. McMahon fails to teach returning a null pointer if no free segment is allocated. RedHat teaches returning a null pointer if no free segment is allocated ("NULL if the request fails", RETURN VALUES). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine McMahon and RedHat to return a null pointer if no free segment is allocated because RedHat describes the function of malloc() which McMahon discusses (Col. 12, lines 52-54).

77. Claims 12-16 and 19-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over McMahon et al. (US 5,784,699) in view of Morzy et al. ("Hierarchical Bitmap Index: An Efficient and Scalable Indexing Technique for Set-Valued Attributes").

78. In regards to claim 12, McMahon further teaches that said bitmap comprises a plurality of second stage bitmaps (group bit map indices, 350 and 360, figure 3C), each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps (bits 1 and 6 of master bit map index, figure 3C), and each bit of the second stage bitmap being indicative of whether or not an associated segment is free ("The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size", Col. 7, lines 26-28), and wherein the operation result is an index to one bit of the second stage bitmap and said one bit of the second stage bitmap being indicative of the state of a segment of a size the same as or larger than the requested memory block ("In one embodiment, a bit flag set as a "1" or high logic level indicates that the corresponding slot contains an available memory block", Col. 7, lines 28-30). McMahon fail to teach a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps. Morzy teaches a plurality of third stage bitmaps (index key leaves, figure 1) and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1) in order to scalably and efficiently index large collections (page 250, paragraph 4). It would have been obvious to a person of

ordinary skill in the art at the time the invention was made to combine McMahon and Morzy for a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps in order to scalably and efficiently index large collections (Morzy, page 250, paragraph 4).

79. In regards to claim 13, McMahon further teaches that if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no more significant bit of said predetermined number of bits of the third stage bitmap (step 145, figure 2).

80. In regards to claim 14, McMahon further teaches that if no free segment is found, repeating the determination for the next more significant bit of said predetermined number of bits of the third stage bitmap, until a free segment is found or there is no more significant bit of said predetermined number of bits of the third stage bitmap (yes arrow of step 150, figure 2).

81. In regards to claim 15, McMahon further teaches that if no free segment is found, repeating the determination for the second stage bitmap associated with the next more significant set bit of the root bitmap, until a free segment is found or there is no more significant bit of the root bitmap (no arrow of step 150, figure 2).

82. In regards to claim 16, McMahon further teaches that each bit of the third stage bitmaps is associated with an entry in a table of pointers indicative of memory addresses of free memory segments ("If a memory block is available for the appropriate

bin size, then the dynamic memory allocator 50 assigns a memory block utilizing the pointer on the free list as shown in blocks 130 and 135", Col. 5, lines 32-35).

83. In regards to claim 19, McMahon further teaches that the bitmap comprises a root bitmap (310, figure 3B), each level being associated with one bit of the root-bitmap (see arrows, figure 3B), and a plurality of second stage bitmaps associated with the segments (groups, 315-330, figure 3B), each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps ("The bit flags in each group bit map index indicate availability of a memory block for a corresponding block size.", Col. 7, lines 26-30). McMahon fails to teach a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps. Morzy teaches a plurality of third stage bitmaps (index key leaves, figure 1) and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1) in order to scalably and efficiently index large collections (page 250, paragraph 4). It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine McMahon with Morzy for a plurality of third stage bitmaps and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps in order to scalably and efficiently index large collections (Morzy, page 250, paragraph 4).

84. In regards to claim 20, Morzy further teaches that the bitmap comprises a root bitmap (index key root level, figure 1), each level being associated with one bit of the root-bitmap (arrows from index key root level, figure 1), and a plurality of second (inner nodes, figure 1) and third stage bitmaps (index key leaves, figure 1) associated with the segments, each bit of the root bitmap being indicative of the state of the bits of an associated one of said second stage bitmaps ("In the root of the index key only first and third bits are set to '1', which means that only first and third inner nodes at the level 2 are non-empty.", page 243, paragraph 1), and each bit of said second stage bitmaps being indicative of the state of an associated predetermined number of bits of one of said third stage bitmaps ("At the upper level (level 2) 4 bits representing non-empty leaf nodes are set to '1'.", page 243, paragraph 1).

85. In regards to claim 21, McMahon further teaches updating the bitmap when a segment is allocated ("a bit map index is maintained to identify available memory blocks", Col. 4, lines 38-39).

86. In regards to claim 22, McMahon further teaches updating the bitmap when a segment is freed ("a bit map index is maintained to identify available memory blocks", Col. 4, lines 38-39).

Conclusion

87. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Sweeny ("Scalability in the XFS File System") teaches allocation groups and extent trees. Masmano et al. ("TLSF: a New Dynamic Memory Allocator for Real-Time Systems") teaches a two-level bitmap allocation scheme with a linked list for

combining blocks. Ogasawara ("An Algorithm with Constant Execution Time for Dynamic Storage Allocation") teaches a hierarchical allocation scheme. Weil ("Leveraging Intra-object Locality with EBOFS") teaches allocating extents. Johnstone ("Non-Compacting Memory Allocation and Real-Time Garbage Collection") teaches a doubly linked list for combining blocks. Burns et al. (US 2003/0014583) teaches a hierarchical allocation scheme. Zbikowski et al. (US 5,713,002) teaches a modified buddy system with bitmaps. Cabrera et al. (US 5,802,599) teaches a doubly linked list for managing free blocks. Ganapathy et al. (US 6,182,089) teaches a hierarchical allocation scheme. Bonola (US 2001/0011338) teaches a binary buddy system. Matsumoto et al. (US 5,517,632) teaches a hierarchical allocation scheme. Forin et al. (US 6,640,290) teaches a hierarchical allocation scheme using bitmaps. Fujii et al. (US 2001/0018731) teaches a three-level hierarchical allocation scheme using bitmaps. Kim et al. (US 6,931,507) teaches a hierarchical allocation scheme. Kuiper et al. (US 6,324,631) teaches a bitmap for combining blocks. Stoney (US 6,505,283) teaches multiple linked lists for managing free space. Atai-Azimi (US 6,845,427) teaches a hierarchical allocation scheme. Li et al. (US 2003/0028739) teaches a hierarchical bitmap allocation scheme.

88. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nathan Sadler whose telephone number is (571)270-7699. The examiner can normally be reached on Monday - Friday 7:30-5:00 EST.

89. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, David Robertson can be reached on (571)272-4186. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

90. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/N. S./
Nathan Sadler
Examiner, Art Unit 4123
May 20, 2009

/David L. Robertson/
Supervisory Patent Examiner
Art Unit 4123